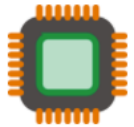


CASPER

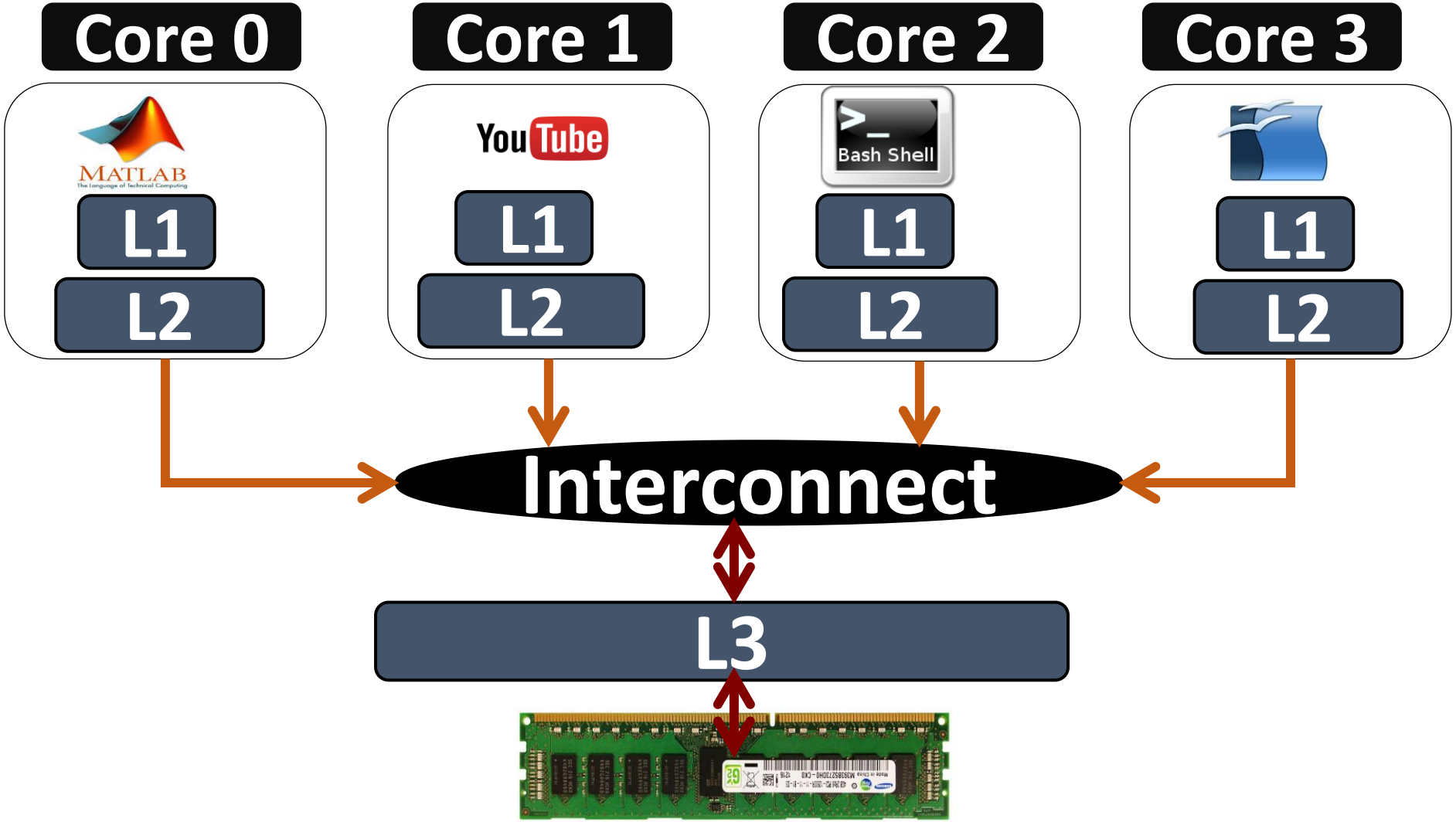
CS305: Computer Architecture

Caches in Multicore

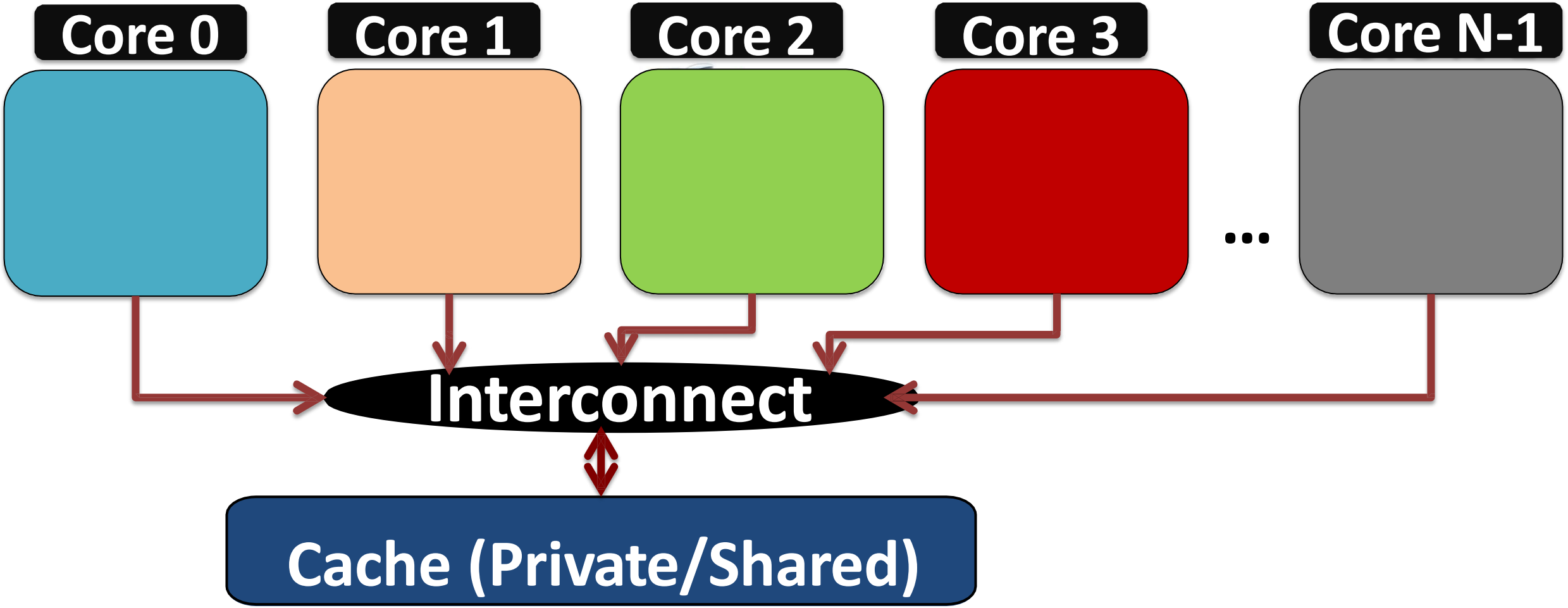
https://www.cse.iitb.ac.in/~biswa/courses/CS305/main.html

https://www.cse.iitb.ac.in/~biswa/

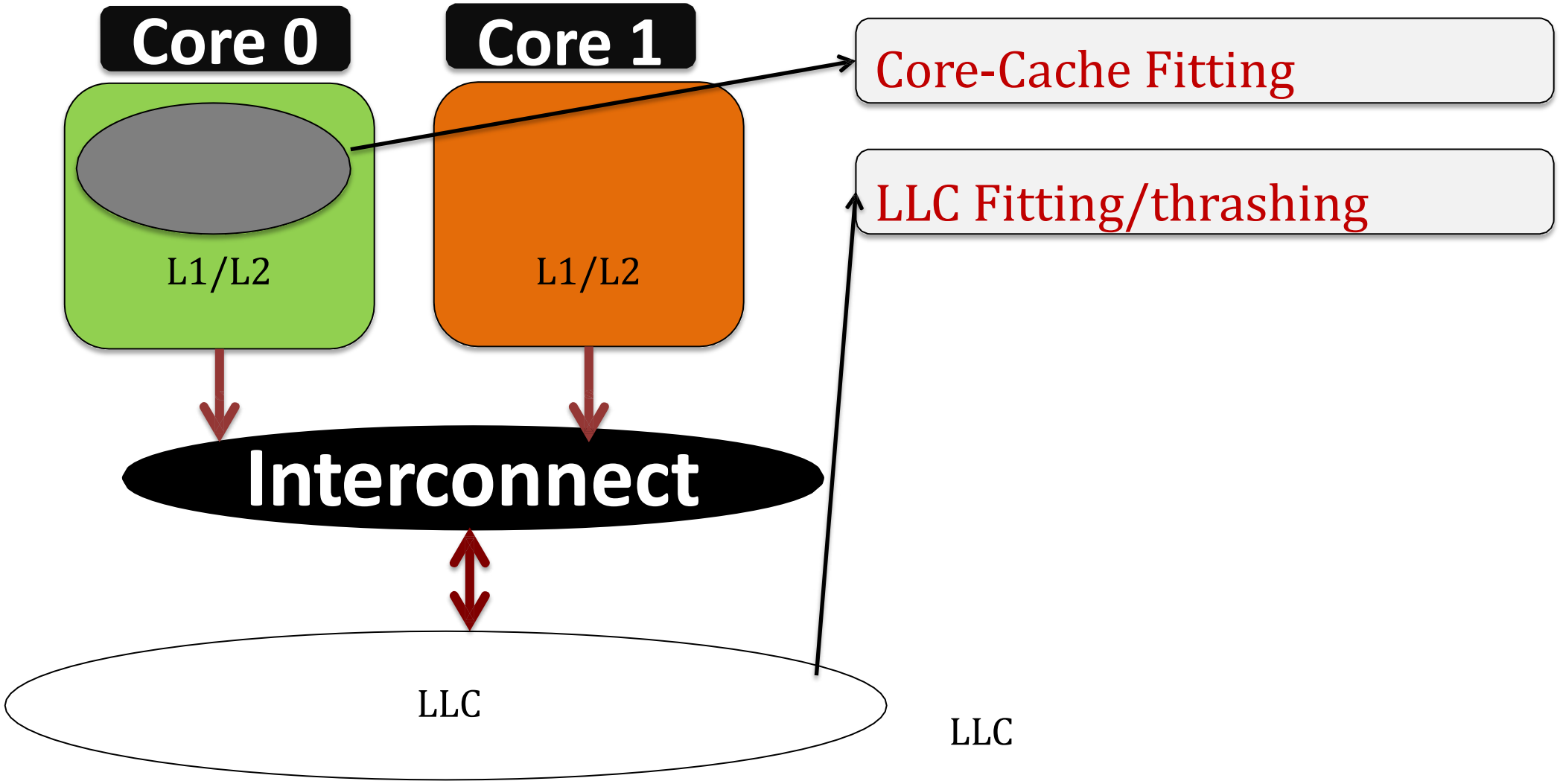
Multicore


Core 0
Core 1
Core 2
Core 3
MATLAB
YouTube
Bash Shell
L1
L2
L1
L2
L1
L2
L1
L2
Interconnect
L3

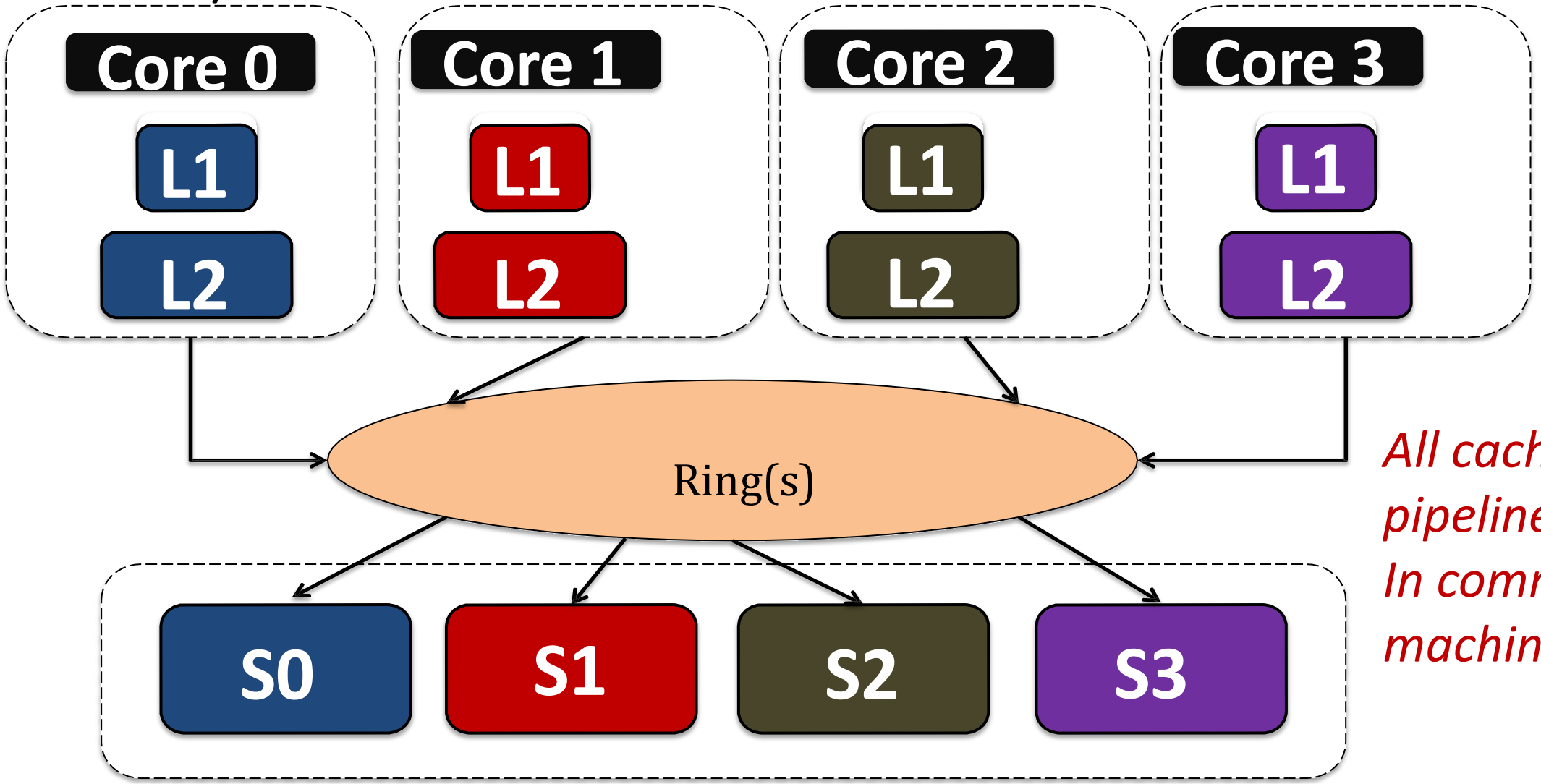
Caches: Private/Shared



Application behavior

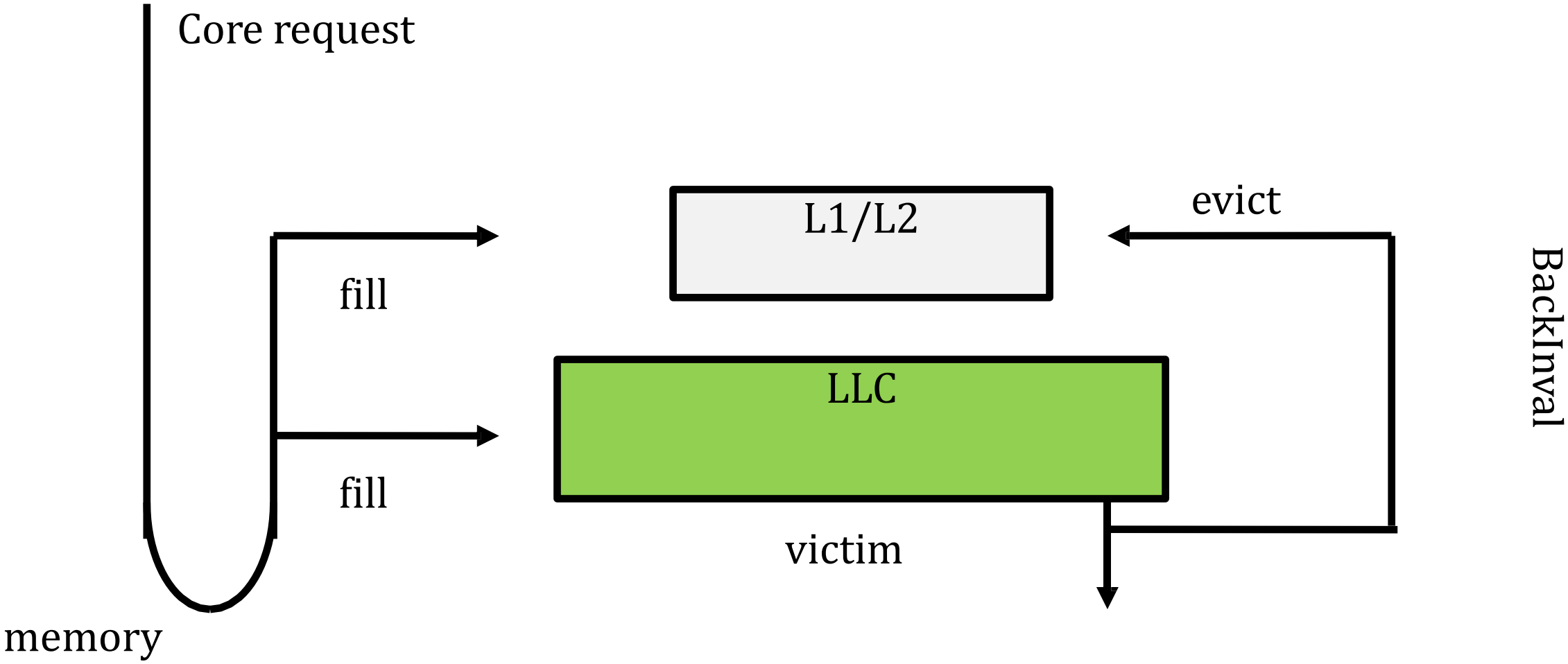


Sliced/Banked LLC

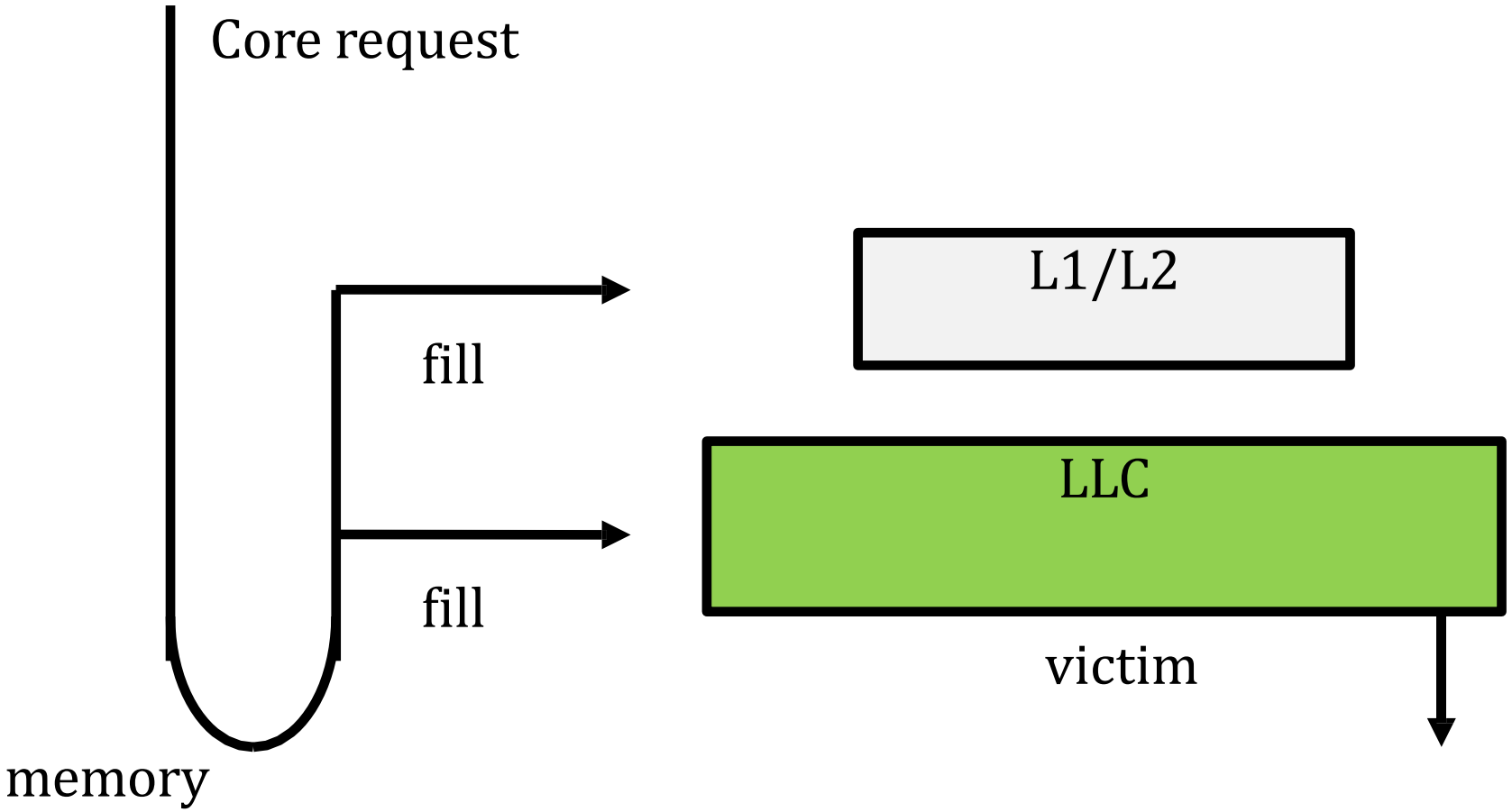


All caches are pipelined
In commercial machines

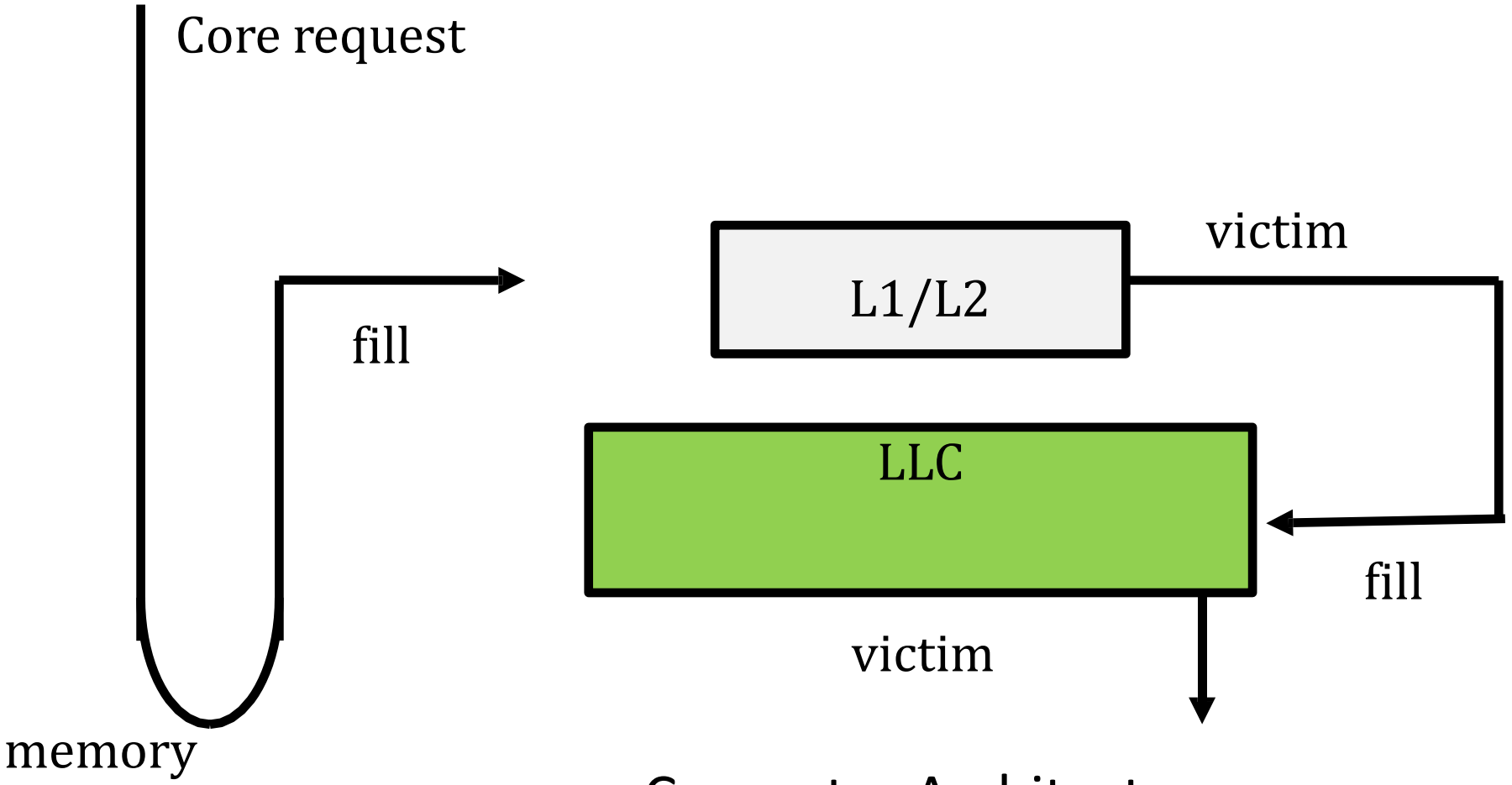
Inclusive Cache Hierarchy



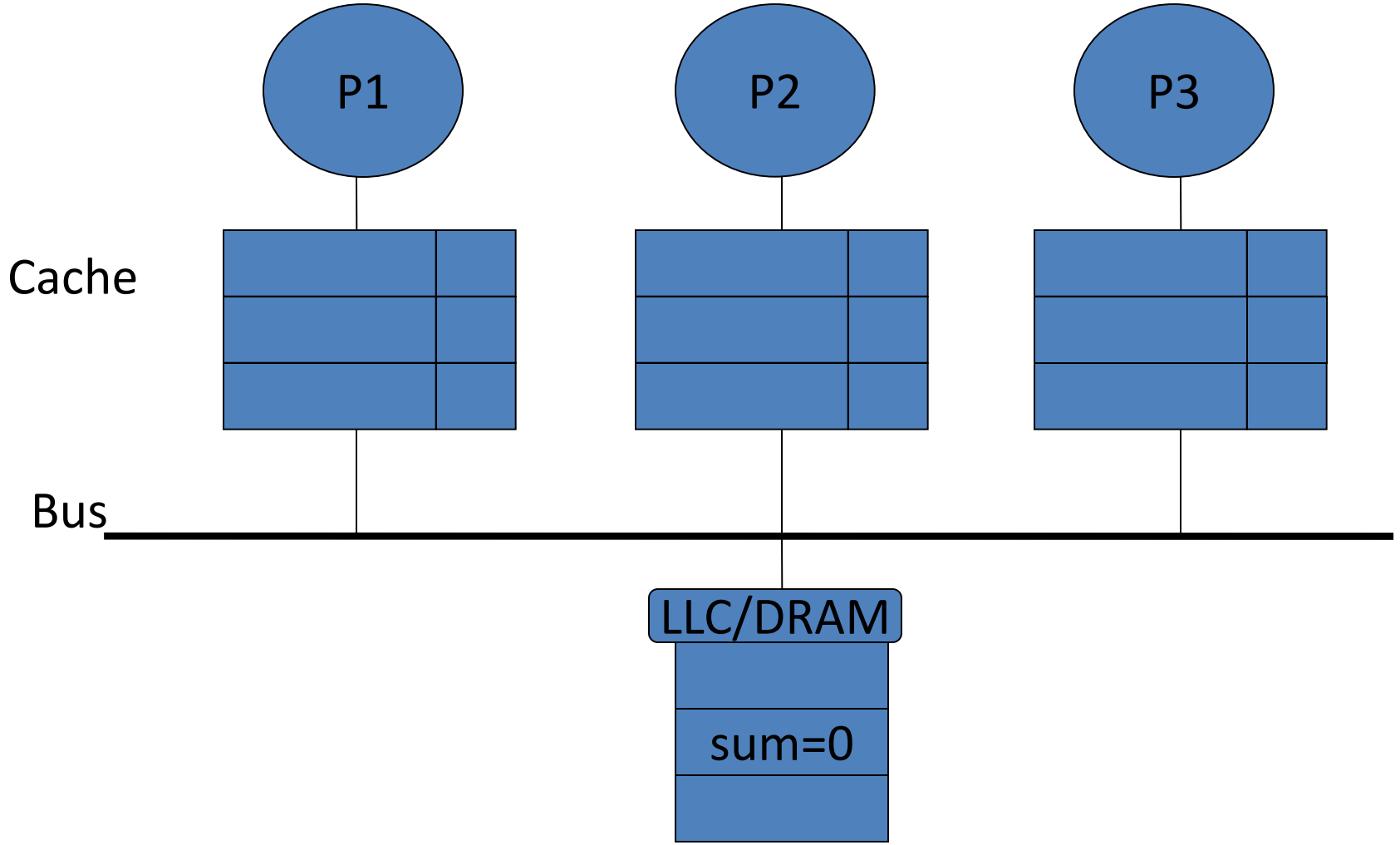
Non-inclusive (many commercial machines)



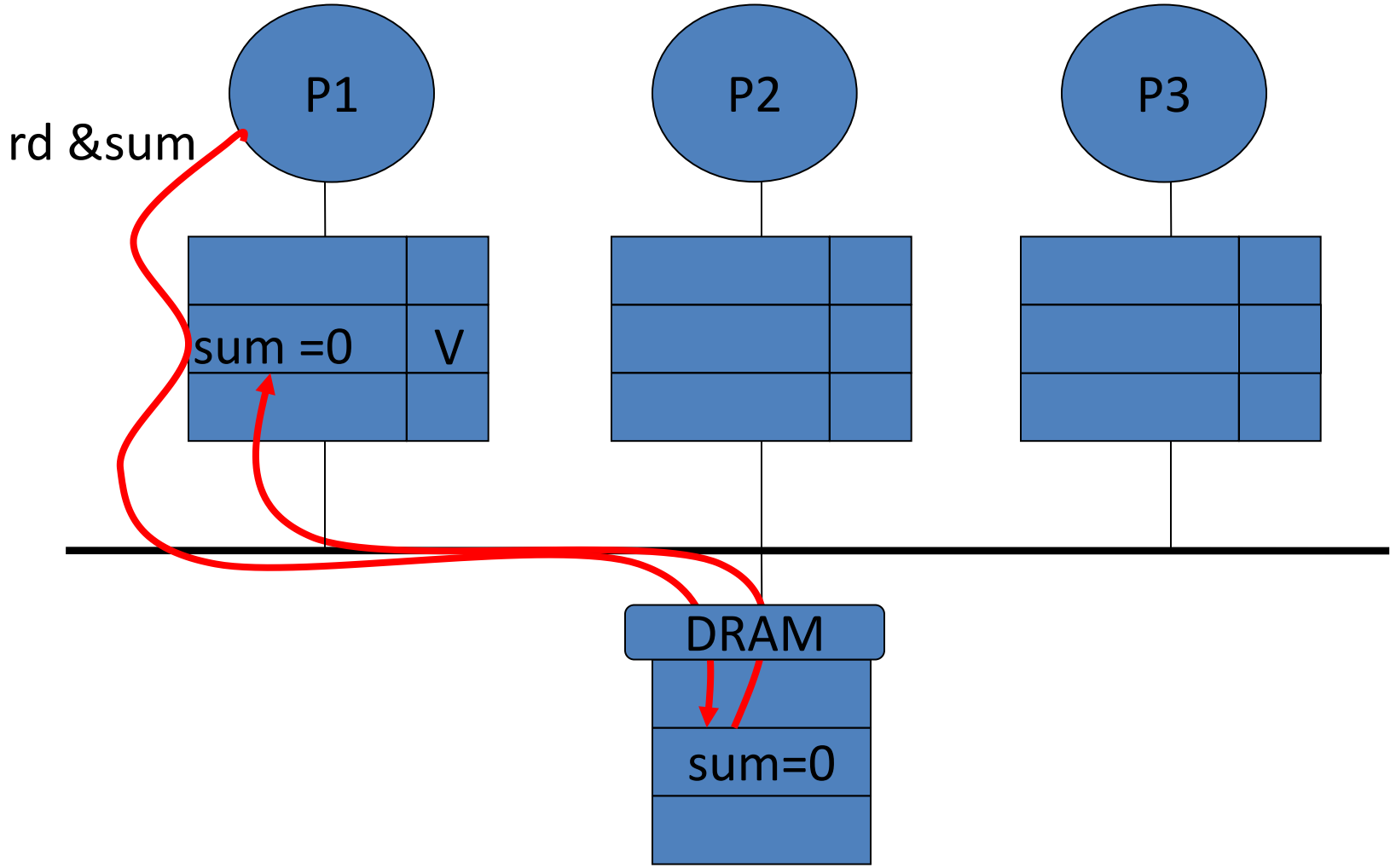
Exclusive hierarchy



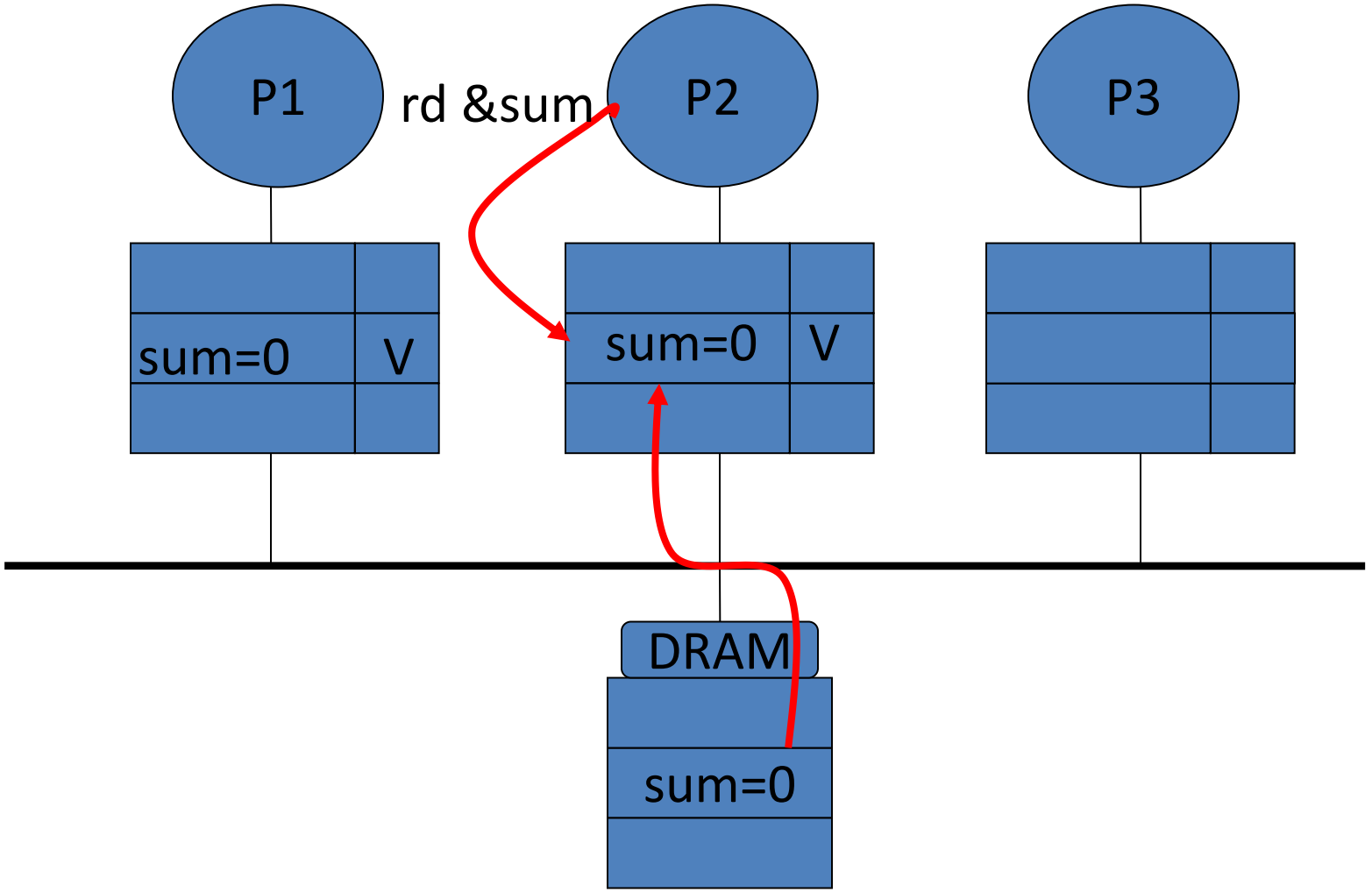
Cache Coherence



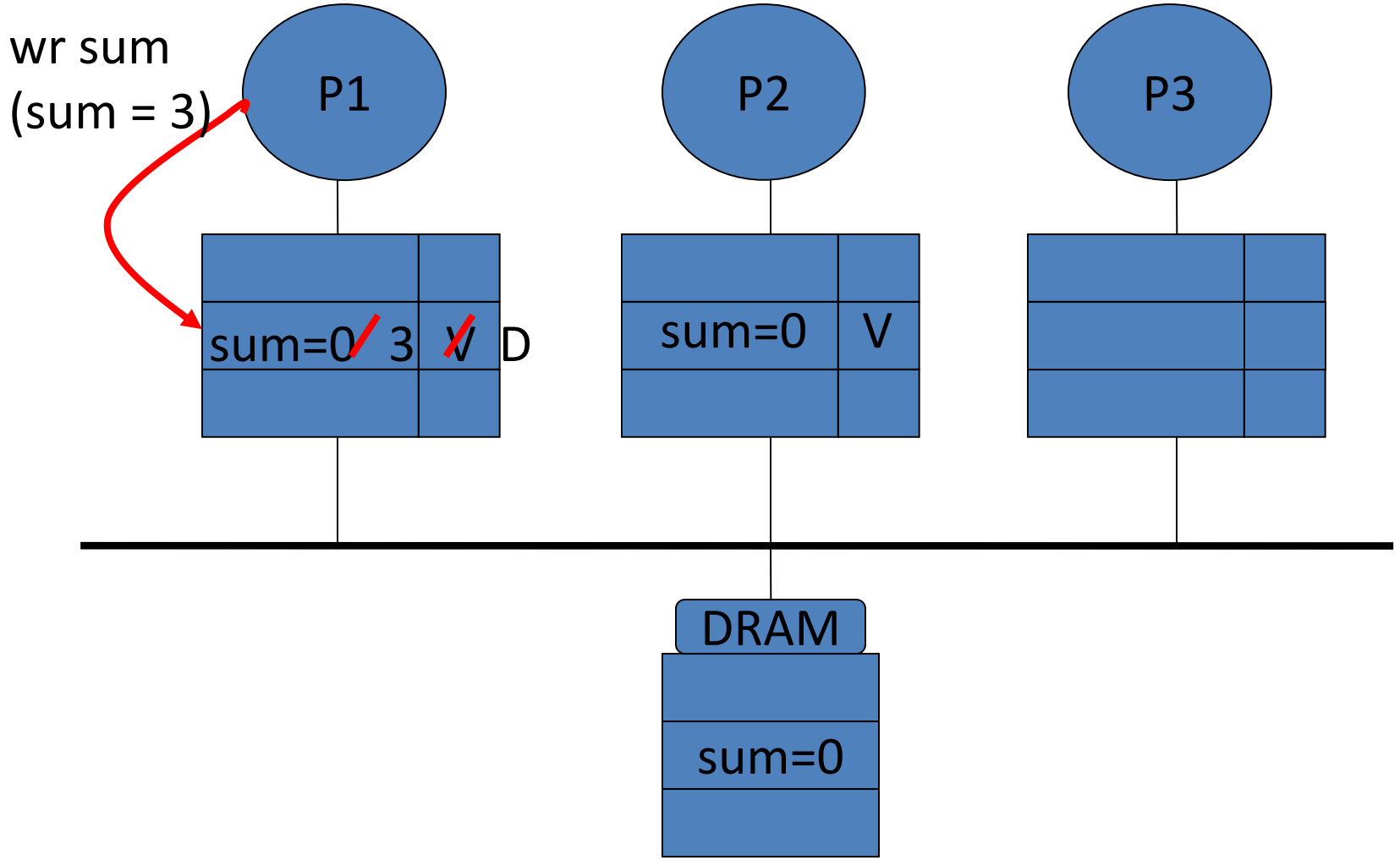
Cache Coherence



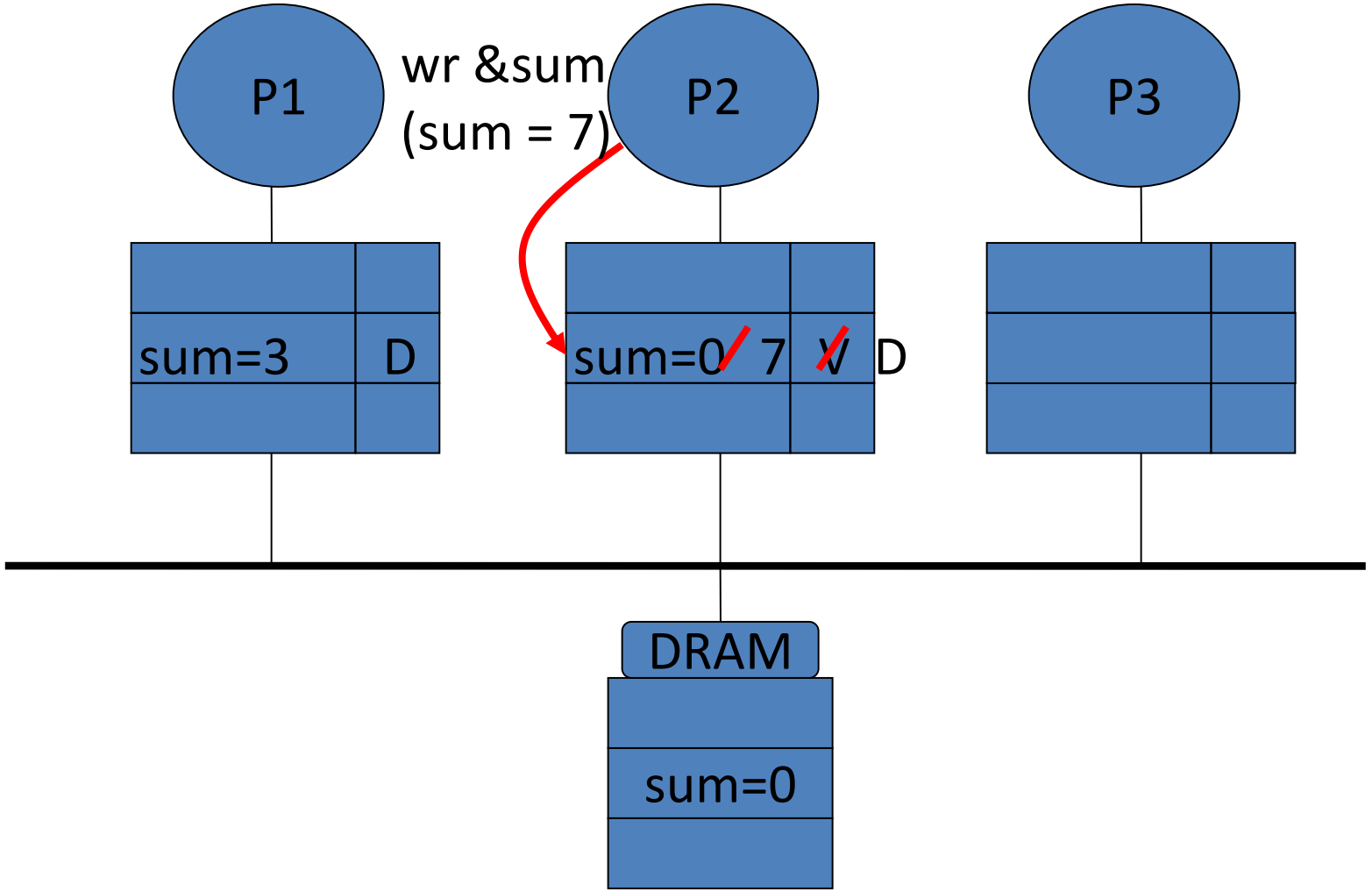
Cache Coherence



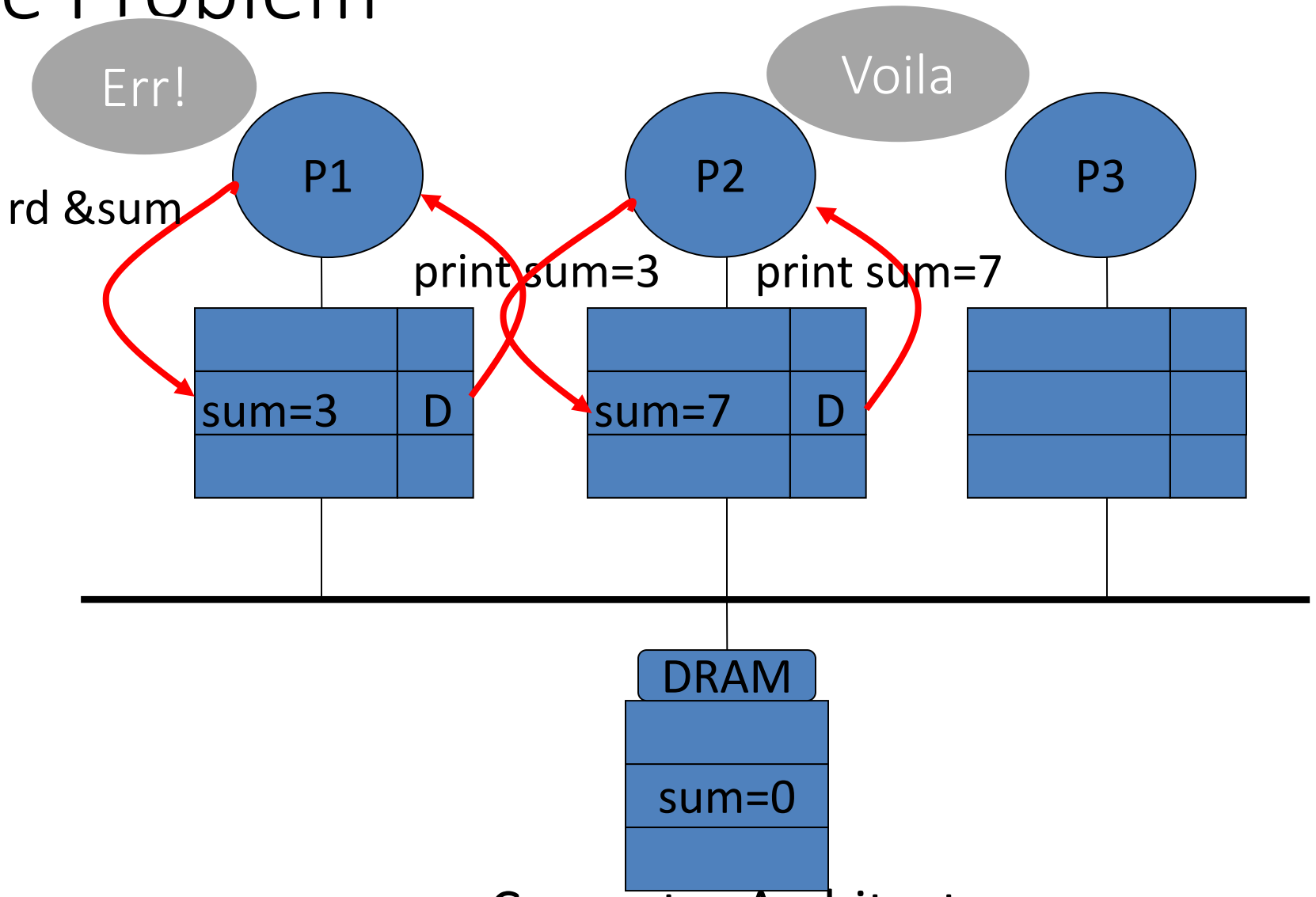
Cache Coherence



Cache Coherence



The Problem


Err!
Voila
P1
P2
P3
rd &sum
print sum=3
print sum=7
sum=3
D
sum=7
D
DRAM
sum=0

The Problem

If multiple cores cache the same block, how do they ensure they all see a consistent state?

Solution:

1. **Write Propagation**: All writes eventually become visible to other cores.
2. **Write Serialization**: All cores see write to a cache line in same order.

Need a protocol to ensure (1) and (2) called *cache coherence protocol*

Invalidate/Update

Invalidate –

- Write to a cache line, and simultaneously broadcast invalidation of address to all others
- Other cores clear/invalidate their cache lines

Update –

- Write to a cache line, and simultaneously broadcast written data to all others
- Other cores update their caches if data was present

A Simple MSI protocol

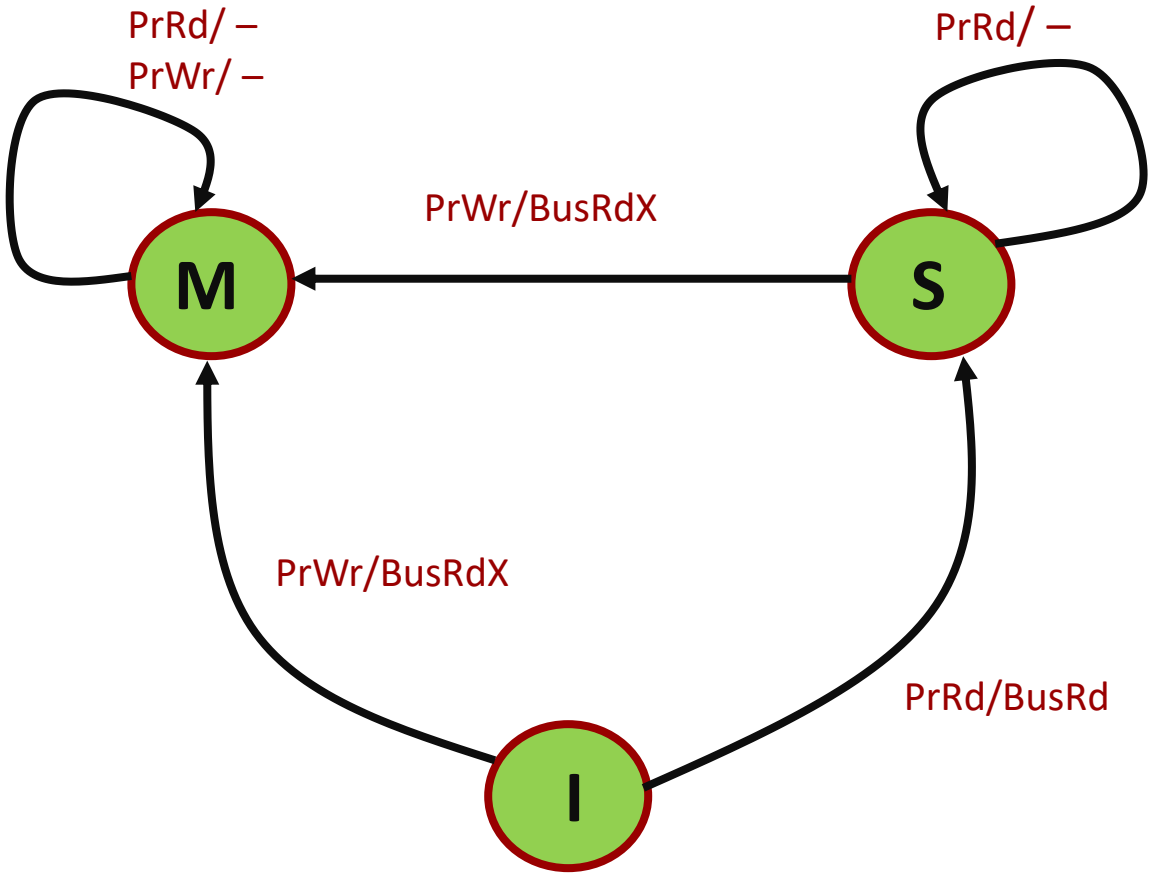
Extend single valid bit per line to three states:

- **M**(odified): only one cache, memory not updated.
- **S**(hared): one or more caches, and memory copy is up-to-date
- **I**(nvalid): not present

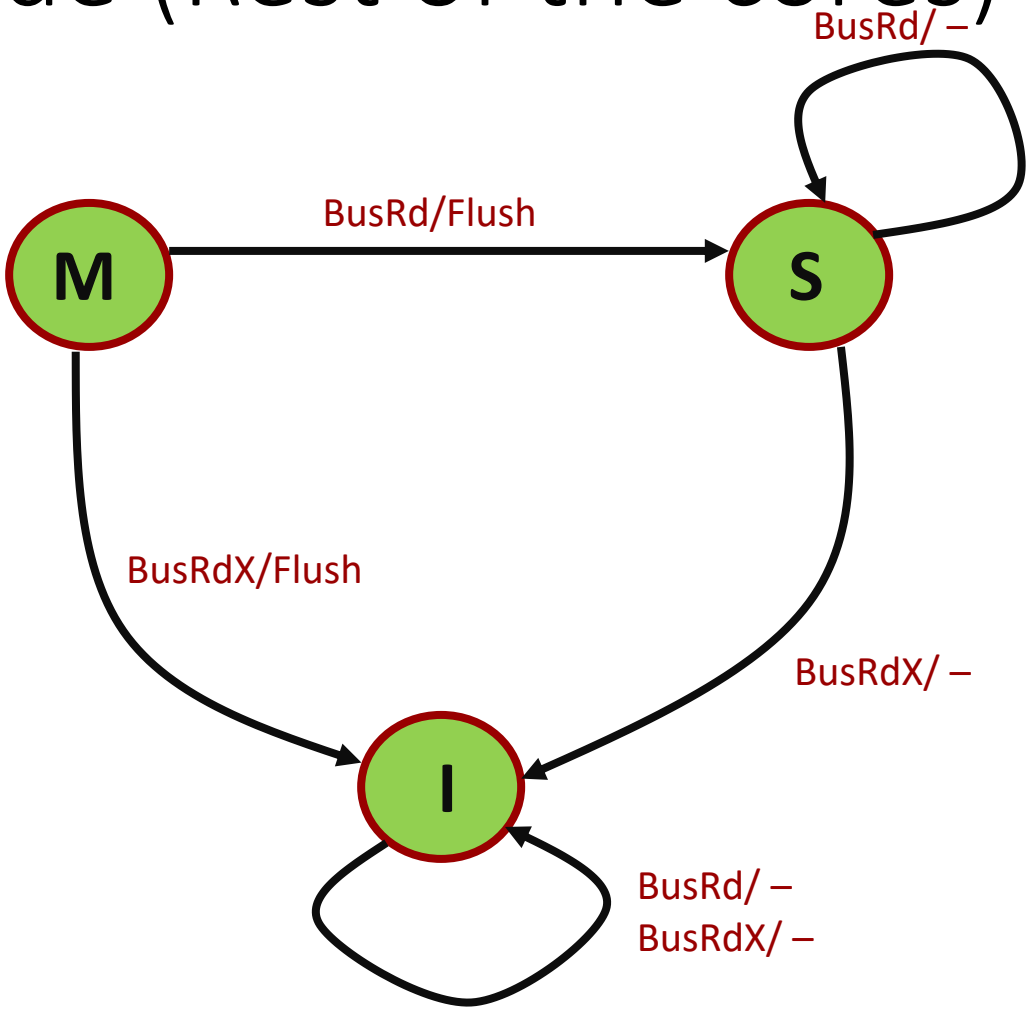
Read - *Read* request on bus, transitions to **S**

Write - *ReadEx* request, transitions to **M** state

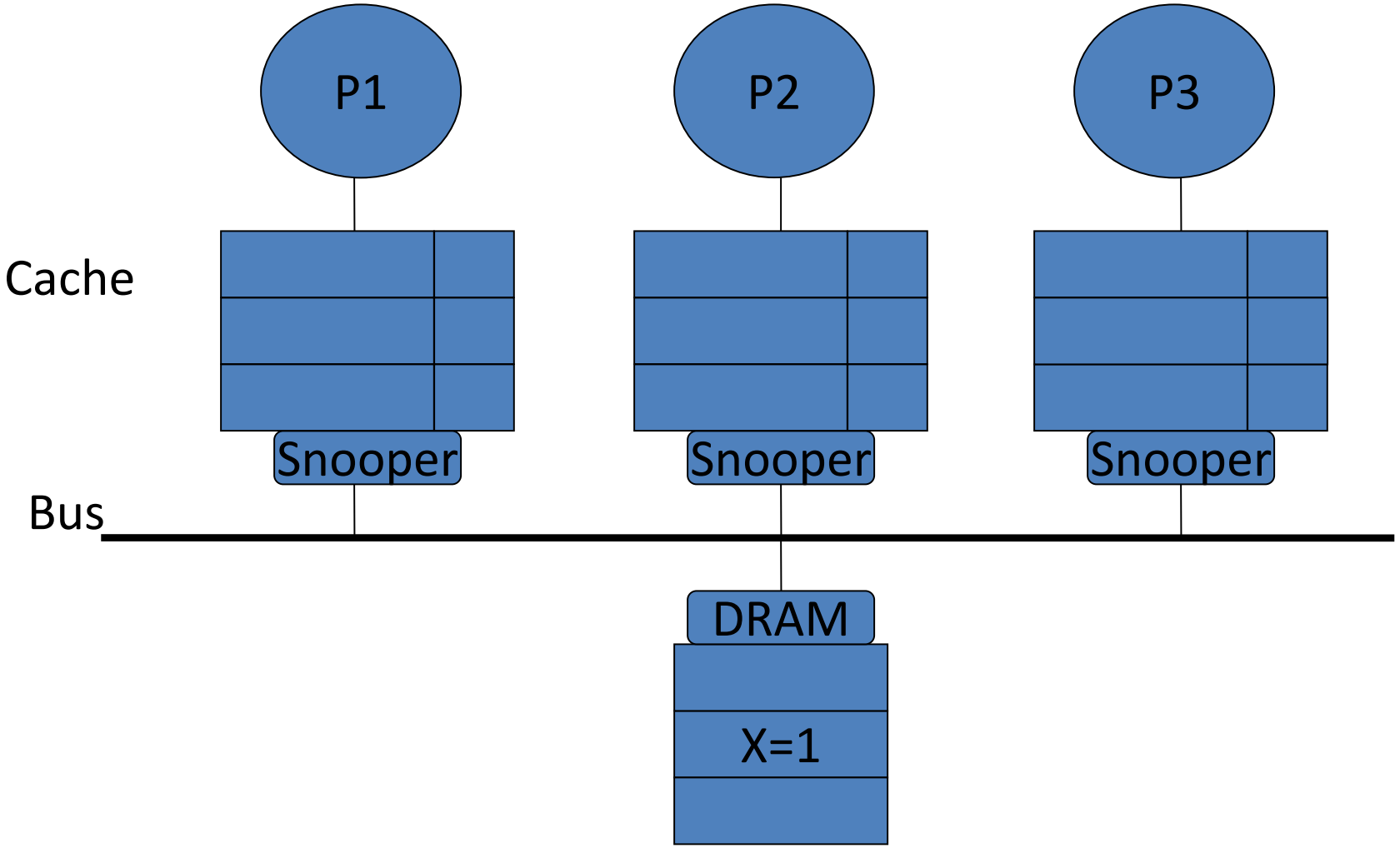
State-transitions: Processor side (Master core)

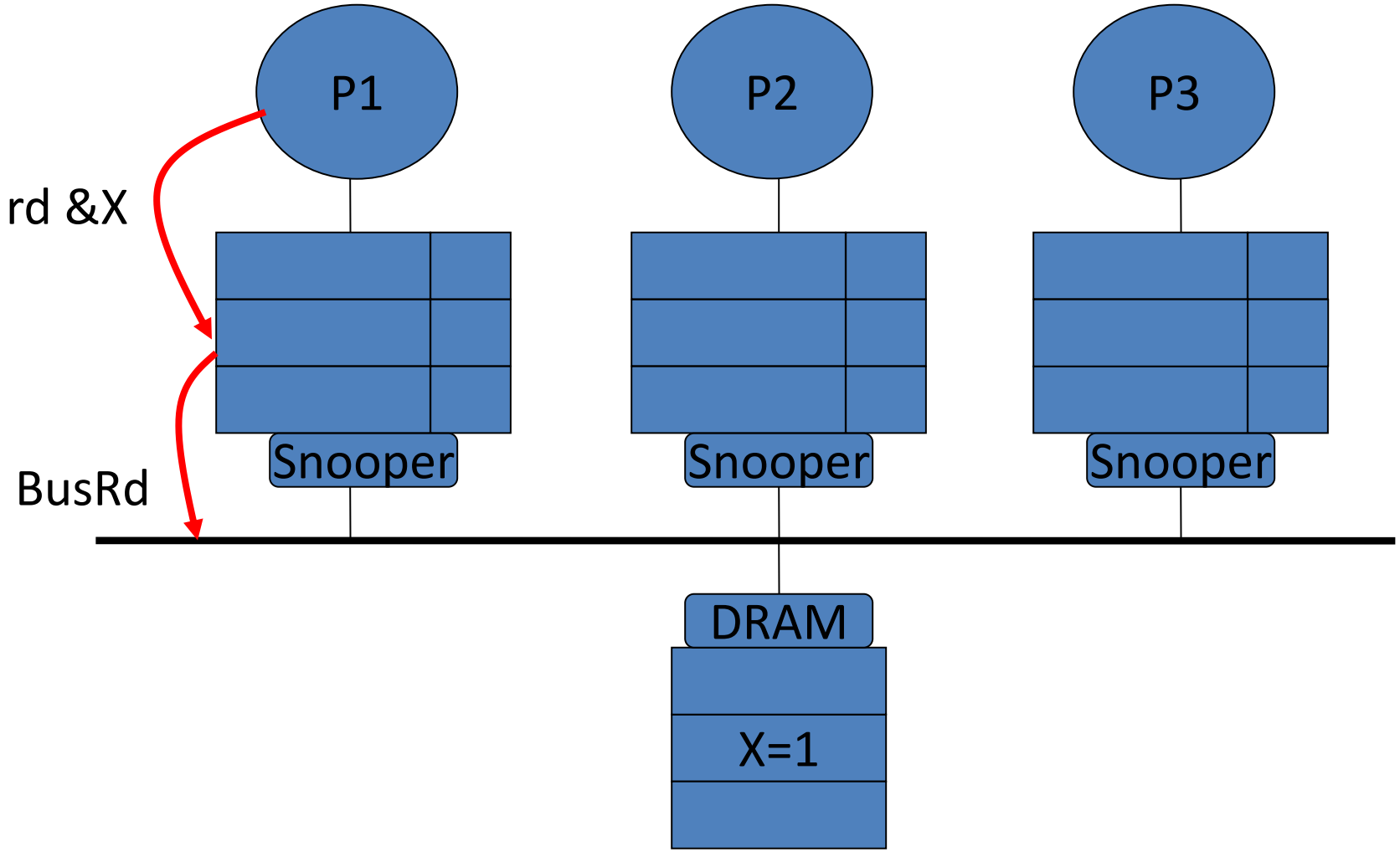


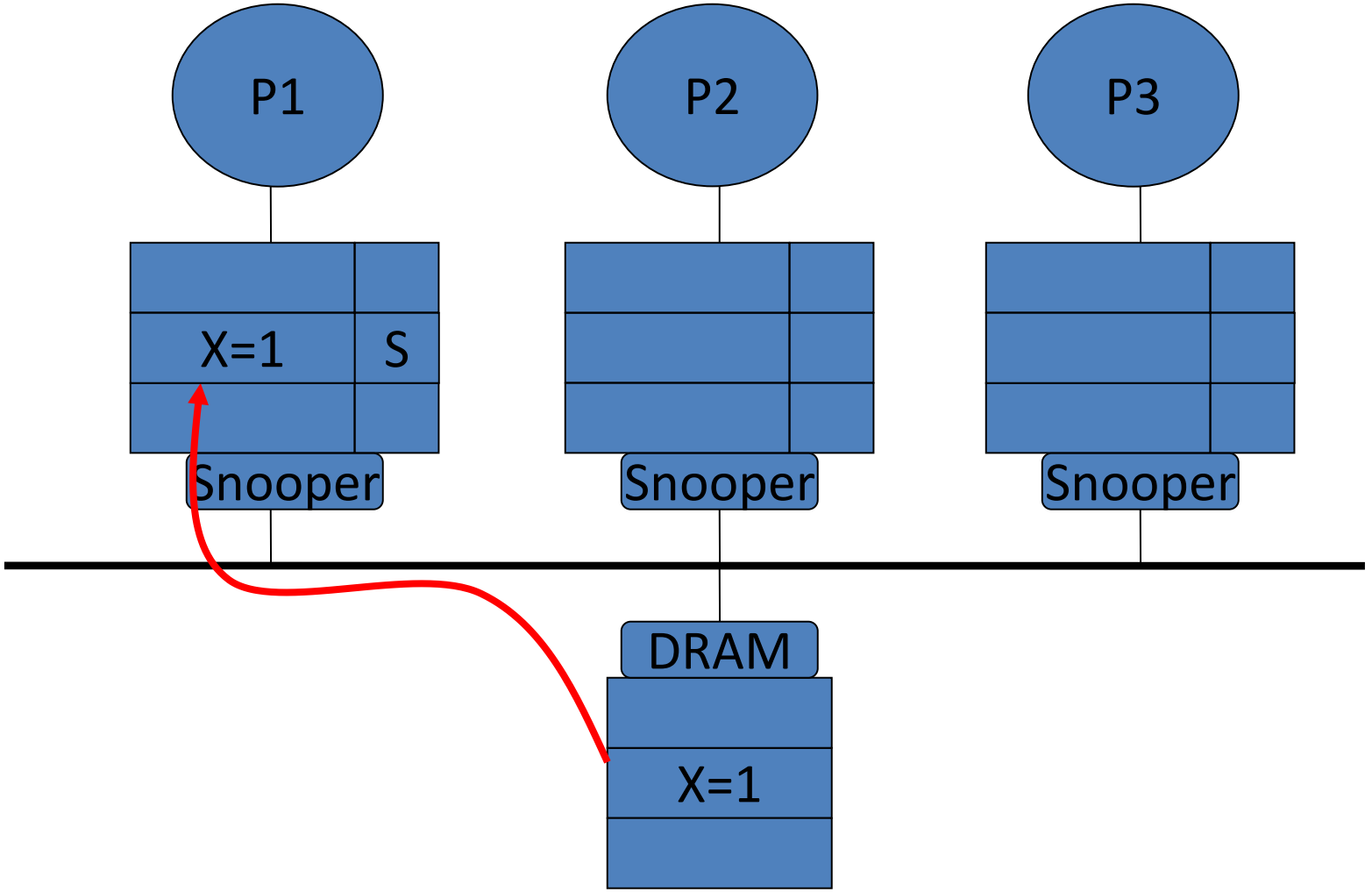
Bus-side (Rest of the cores)

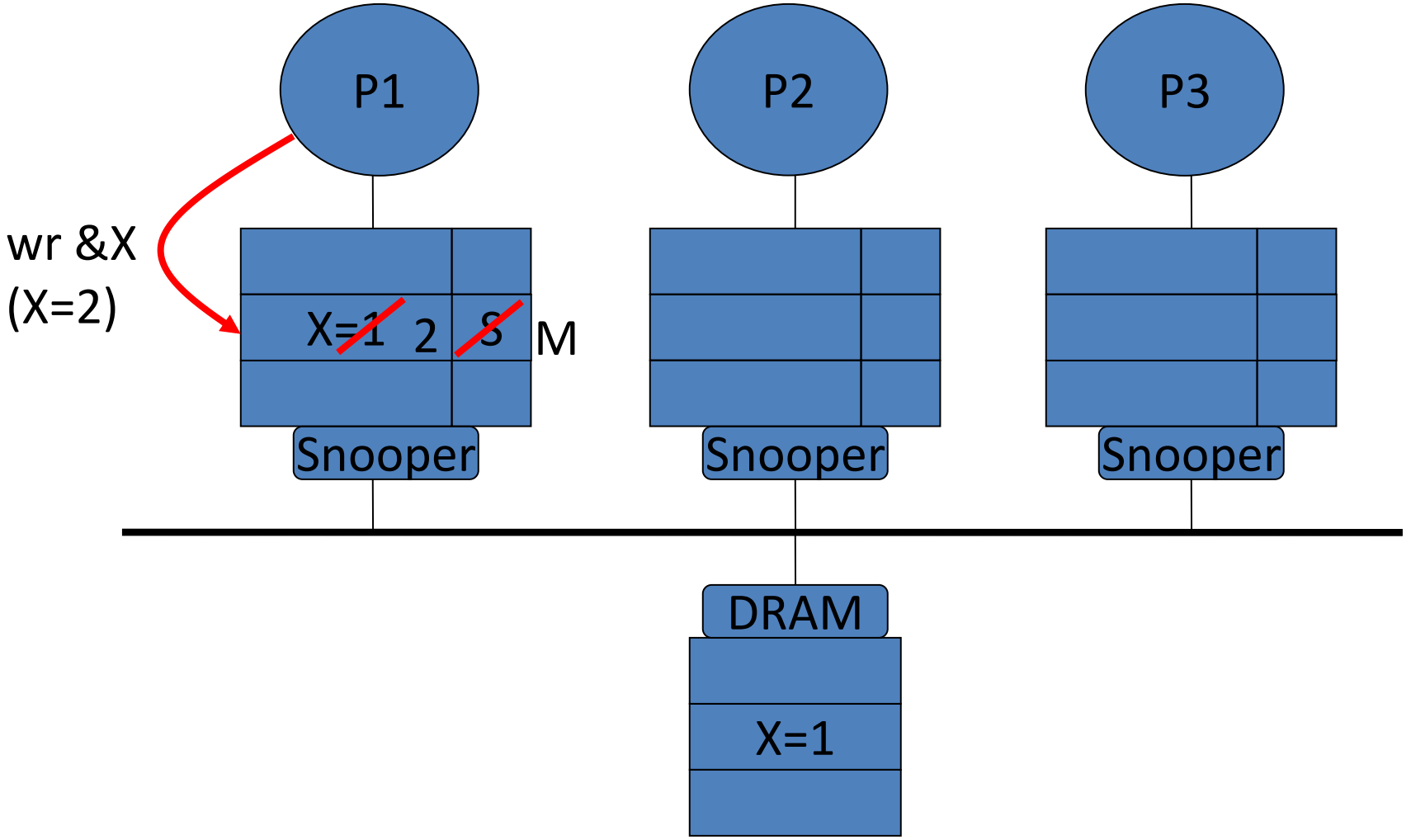


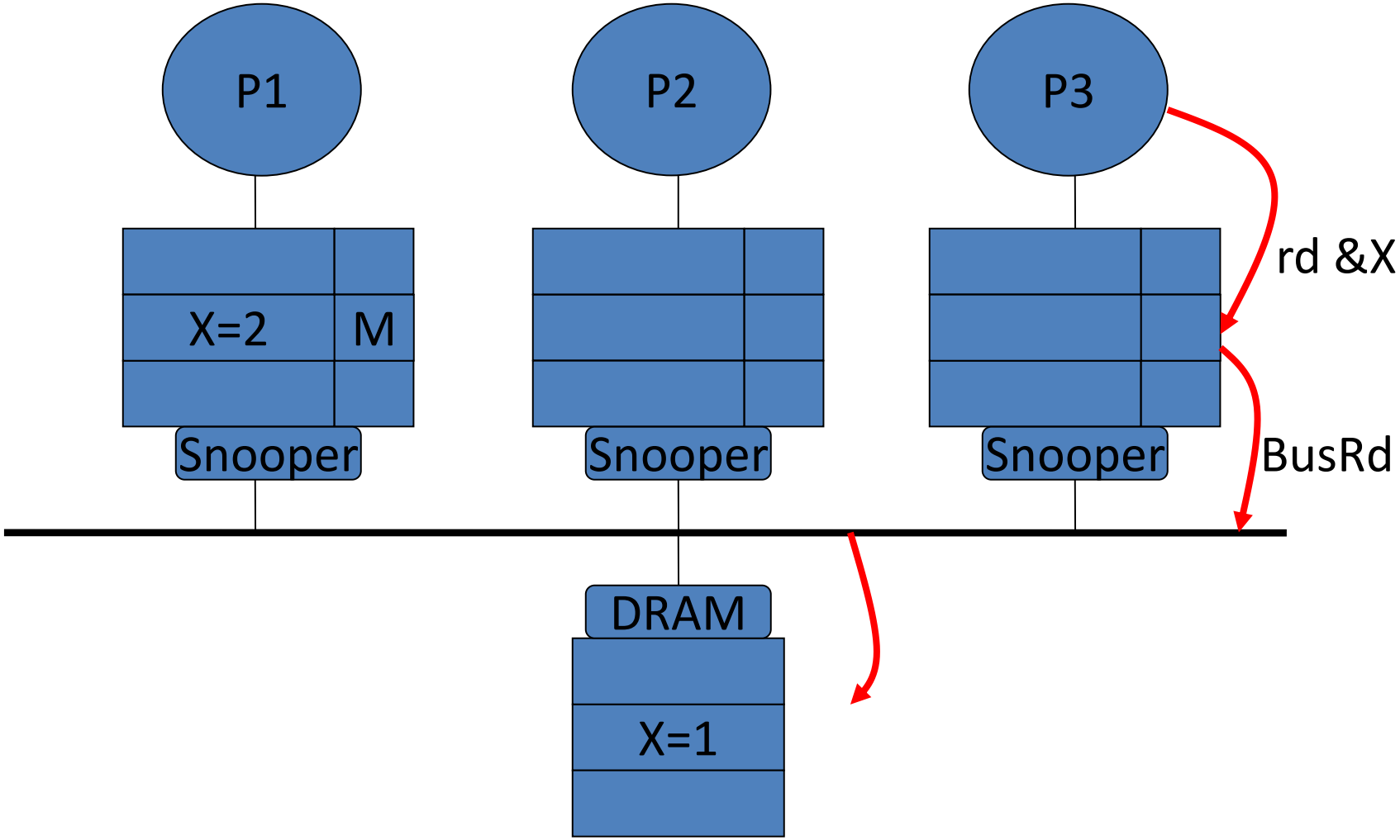
MSI Protocol

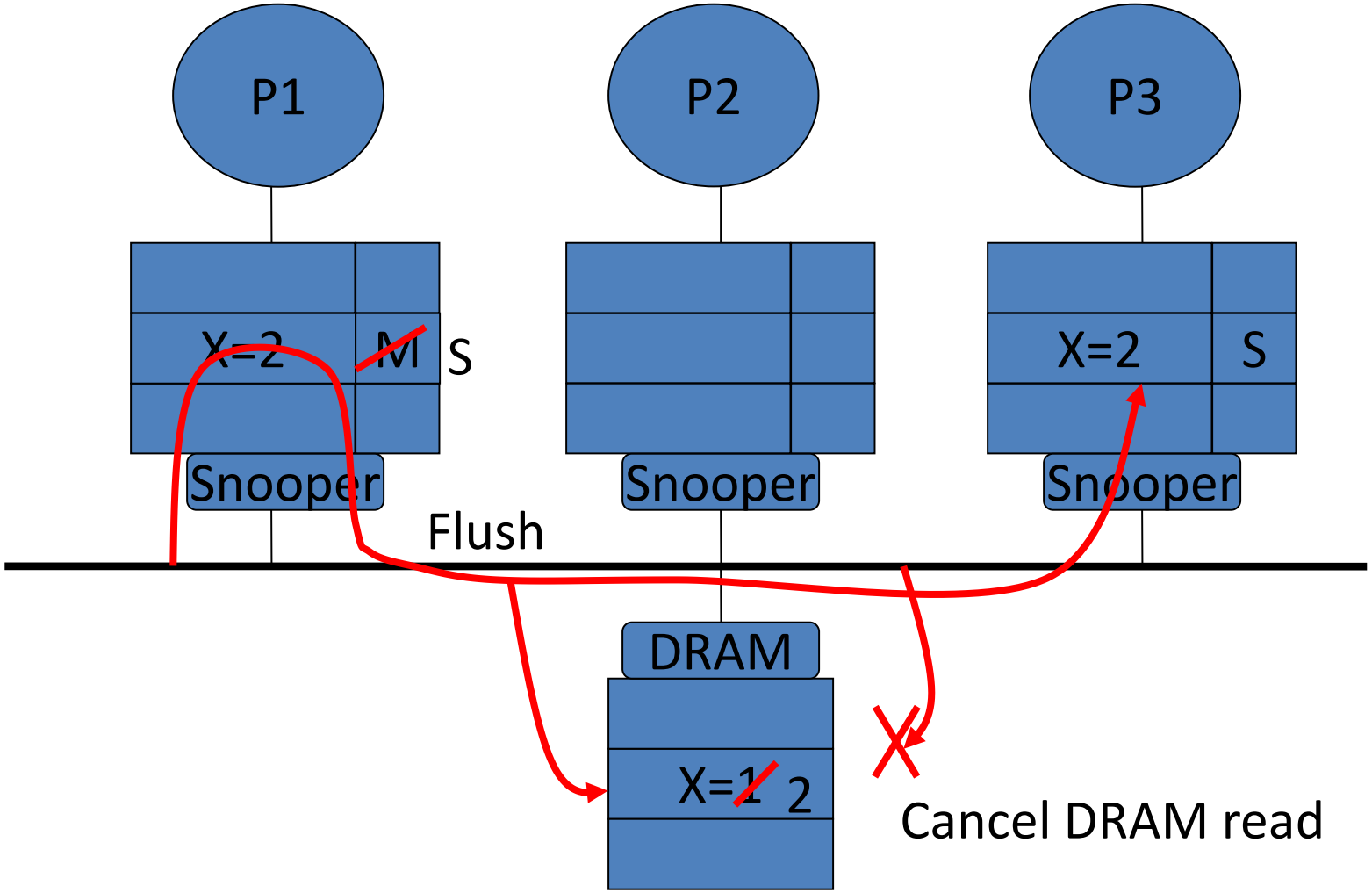


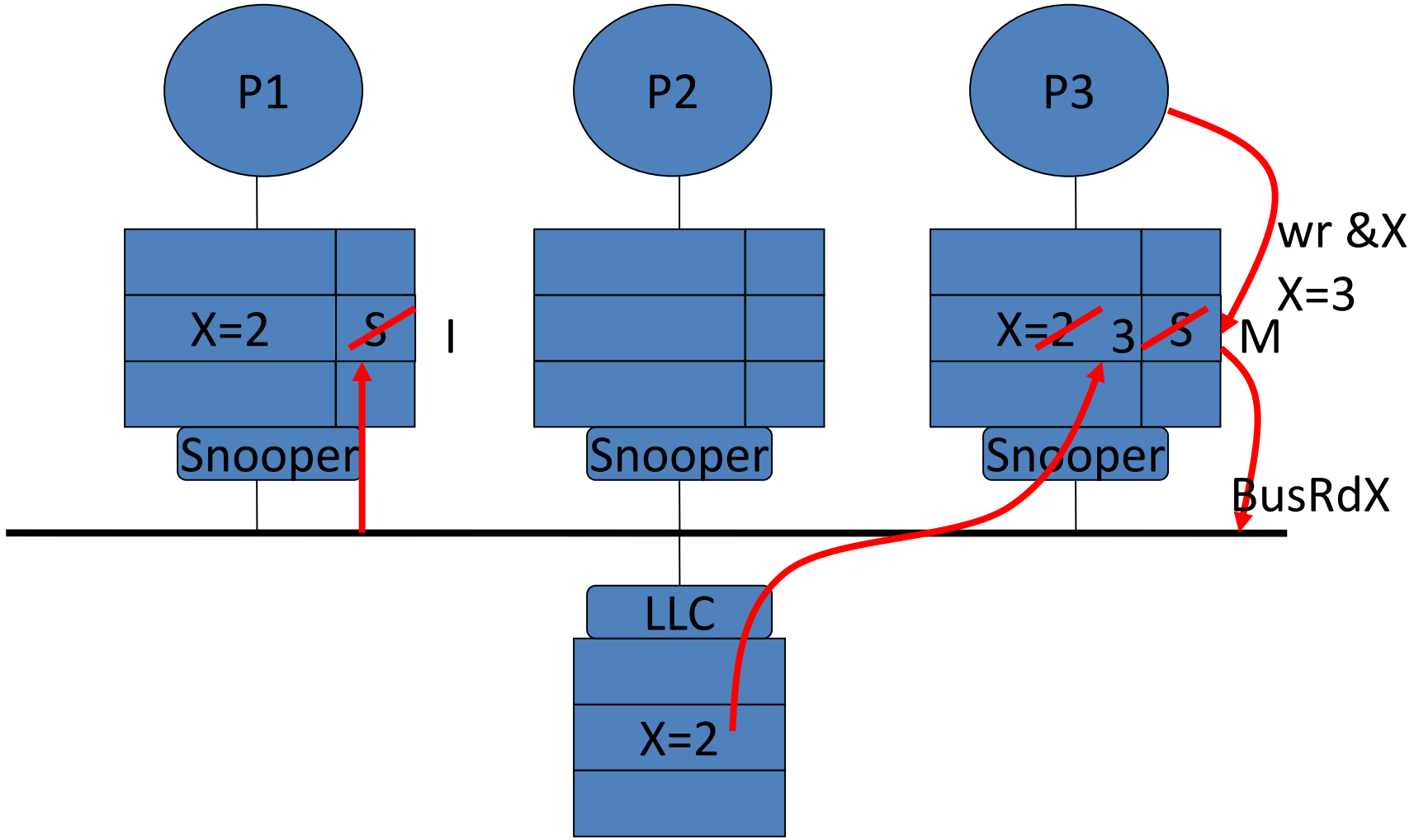

P1
P2
P3
rd &X
Snooper
Snooper
Snooper
BusRd
DRAM
X=1

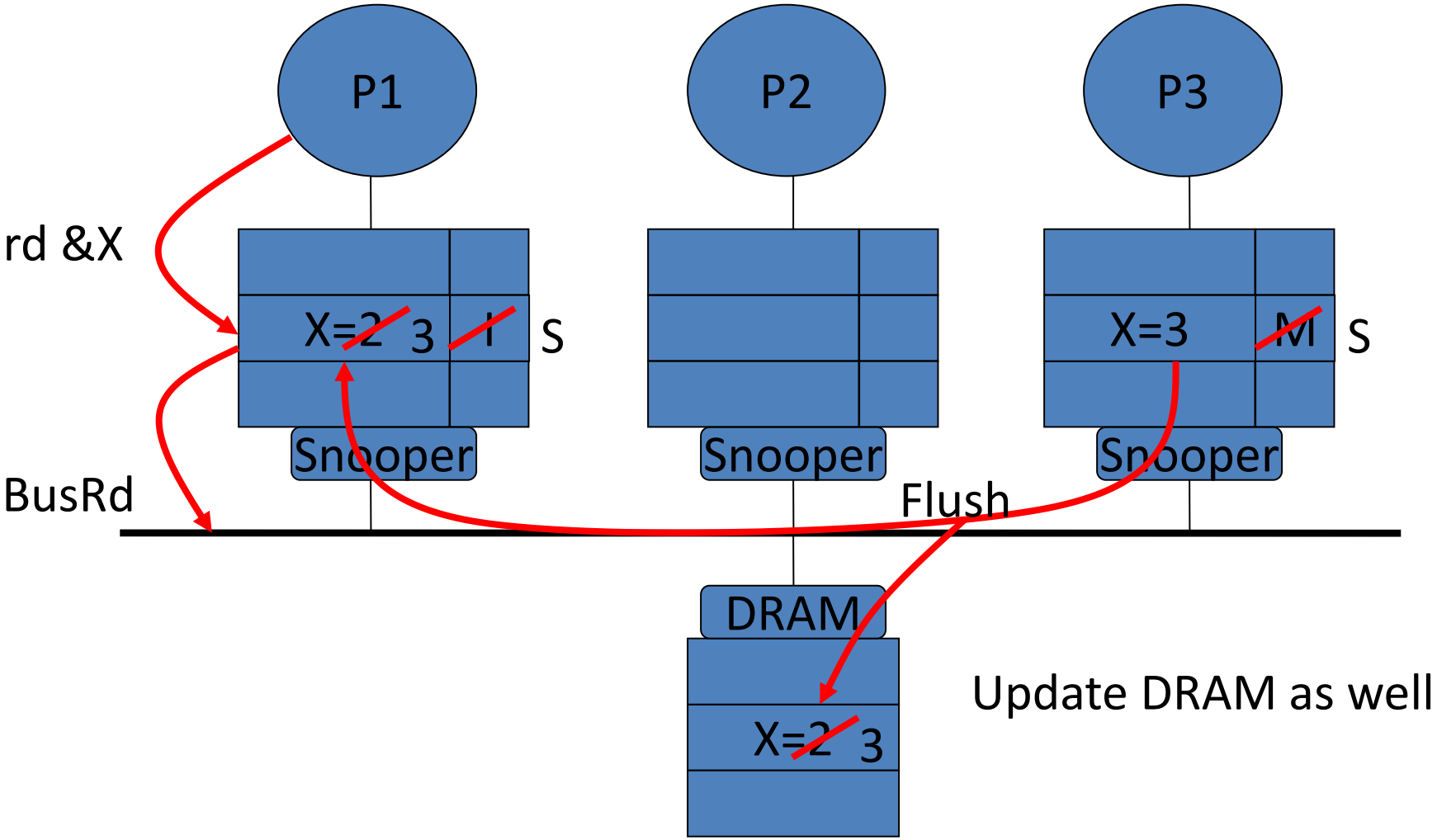

P1
P2
P3
X=1
S
Snooper
Snooper
Snooper
DRAM
X=1

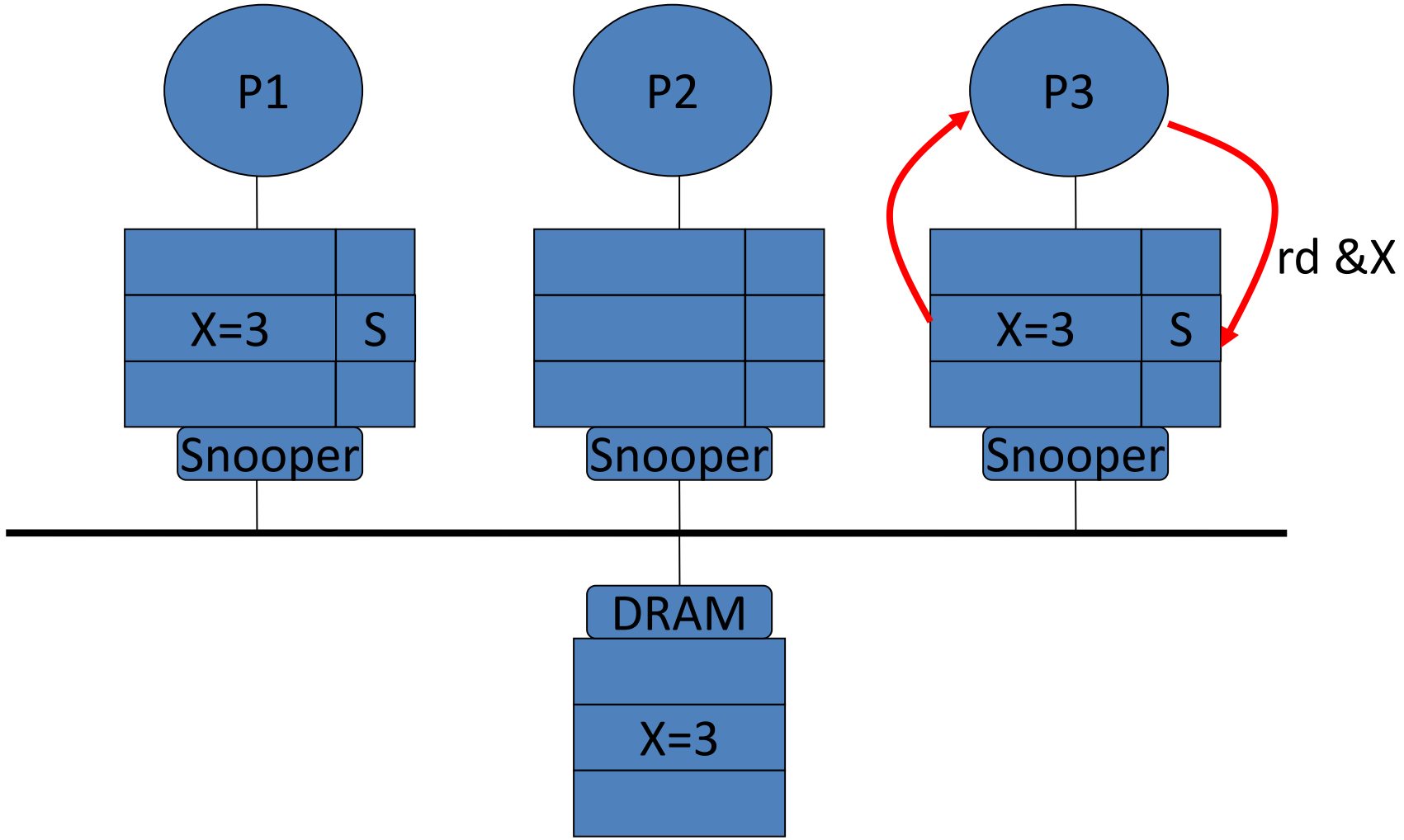

P1
P2
P3
wr &X
(X=2)
X=1
2
S
M
Snooper
Snooper
Snooper
DRAM
X=1


P1
P2
P3
X=2
M
Snooper
Snooper
Snooper
rd &X
BusRd
DRAM
X=1


P1
P2
P3
X=2
M
S
X=2
S
Snooper
Snooper
Snooper
Flush
DRAM
X=1
2
Cancel DRAM read


P1
P2
P3
X=2
S
I
Snooper
Snooper
X=2
3
S
M
Snooper
wr &X
X=3
BusRdX
LLC
X=2


P1
P2
P3
rd &X
X=2
3
I
S
X=3
M
S
Snooper
Snooper
Snooper
BusRd
Flush
DRAM
X=2
3
Update DRAM as well


P1
P2
P3
X=3
S
X=3
S
rd &X
Snooper
Snooper
Snooper
DRAM
X=3

Summary

Proc Action	State P1	State P2	State P3	Bus Action	Data From
R1	-	-	-	BusRd	Mem
W1	-	-	-	BusRdX	Mem
R3	-	-	-	BusRd	P1 cache
W3	-	-	-	BusRdX	Mem
R1	-	-	-	BusRd	P3 cache
R3	-	-	-	-	-
R2	-	-	-	BusRd	Mem

Summary

Proc Action	State P1	State P2	State P3	Bus Action	Data From
R1	S	-	-	BusRd	Mem
W1	M	-	-	BusRdX	Mem
R3	S	-	S	BusRd	P1 cache
W3	I	-	M	BusRdX	Mem
R1	S	-	S	BusRd	P3 cache
R3	S	-	S	-	-
R2	S	S	S	BusRd	Mem

Terima kasih